

(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau



(43) International Publication Date
13 September 2001 (13.09.2001)

PCT

(10) International Publication Number
WO 01/67447 A2

- (51) International Patent Classification⁷: **G11B 20/00** (deceased). **VIS, Marvin, L.**; 3841 Staghorn Drive, Longmont, CO 80503 (US).
- (21) International Application Number: PCT/US01/07616
- (74) Agent: **SHIFRIN, Dan, A.**; Cirrus Logic, Inc., 4210 S. Industrial Drive, Austin, TX 78744 (US).
- (22) International Filing Date: 8 March 2001 (08.03.2001)
- (25) Filing Language: English
- (81) Designated States (*national*): JP, KR.
- (26) Publication Language: English
- (84) Designated States (*regional*): European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE, TR).
- (30) Priority Data:
09/521,554 9 March 2000 (09.03.2000) US
- Published:
— without international search report and to be republished upon receipt of that report
- (71) Applicant: **CIRRUS LOGIC, INC.** [US/US]; 4210 S. Industrial Drive, Austin, TX 78744 (US).
- (72) Inventors: **TURK, Stephen, A.**; 841 Bross Street, Longmont, CO 80501 (US). **ZOOK, Christopher, P.**
- For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.*



WO 01/67447 A2

(54) Title: MULTIPLE-RATE CHANNEL ENDEC IN A COMMUTED READ/WRITE CHANNEL FOR DISK STORAGE SYSTEMS

(57) Abstract: A sampled amplitude read channel is disclosed for writing data to and reading data from a disk storage medium. A first channel encoder encodes a first j-k bits of a j-bit data block to generate first encoded data, and an ECC encoder encodes the first encoded data and a remaining k-bits of the data block to generate ECC redundancy symbols comprising a plurality of bits. A second channel encoder encodes the remaining k-bits of the data block and the ECC redundancy symbols to generate second encoded data. The first encoded data and the second encoded data are then output as channel data written to the disk storage medium.

MULTIPLE-RATE CHANNEL ENDEC IN A COMMUTED READ/WRITE
CHANNEL FOR DISK STORAGE SYSTEMS

FIELD OF INVENTION

The present invention relates to communication systems, such as the recording and reproduction of binary data in disk storage systems for digital computers. In particular, the present invention relates to a commuted read/write channel employing a multiple-rate channel encoder/decoder (ENDEC) which increases storage capacity by reducing or eliminating pad bits required in prior art configurations.

CROSS REFERENCE TO RELATED APPLICATIONS AND PATENTS

This application is related to concurrently filed U.S. Patent Application serial no. __/__,__ entitled "A COST-EFFECTIVE HIGH-THROUGHPUT ENUMERATIVE ENDEC EMPLOYING A PLURALITY OF SEGMENTED COMPARE TABLES." This application is also related to U.S. Patent No. 6,009,549 entitled "DISK STORAGE SYSTEM EMPLOYING ERROR DETECTION AND CORRECTION OF CHANNEL CODED DATA, INTERPOLATED TIMING RECOVERY, AND RETROACTIVE/SPLIT-SEGMENT SYMBOL SYNCHRONIZATION." The above-identified U.S. patent applications and U.S. patents are incorporated herein by reference.

1 BACKGROUND OF THE INVENTION

2 Disk storage system (e.g., magnetic and optical disk drives)
3 are essentially communication systems wherein the storage medium,
4 head and read/write electronics constitute the communication
5 channel. User data are transmitted through a communication medium
6 by recording the data to the disk during a write operation and
7 sensing the recorded data during a read operation.

8 FIG. 1A illustrates the typical format of a magnetic disk
9 storage medium comprising a plurality of concentric, radially
10 spaced data tracks 2 partitioned into a number of data sectors 4.
11 A typical format for a data sector 4 is shown in FIG. 1B as
12 comprising a preamble 3 for acquiring the frequency and phase of
13 the recorded data, a sync mark 5 for symbol synchronizing to the
14 recorded data, channel encoded user data 7, and appended ECC
15 redundancy symbols 9 for detecting and correcting errors in the
16 user data. The disk may also include embedded servo sectors 6
17 which facilitate positioning a head with respect to the disk during
18 write and read operations. In order to achieve a more constant
19 linear bit density, the tracks are typically banded together to
20 form zones and the data rate is increased from the inner to outer
21 diameter zones. This is illustrated in FIG. 1A wherein the disk
22 is partitioned into an inner zone 10 comprising seven data sectors
23 per track and an outer zone 8 comprising fourteen data sectors per
24 track. In practice, the disk is actually partitioned into several
25 zones with the data rate increasing incrementally from the inner

1 to outer diameter zones.

2 The user data are typically encoded in order to increase the
3 effective signal-to-noise ratio (SNR) which facilitates higher
4 linear recording densities leading to an increase in storage
5 capacity. Two types of codes are typically employed: a channel
6 code and an error correction code (ECC). A channel code typically
7 increases the effective SNR by attenuating a noise source within
8 the recording channel. For example, a run-length limited (RLL)
9 channel code constrains the minimum spacing between consecutive
10 medium transitions representing the recorded data which reduces
11 intersymbol interference. An RLL channel code may also constrain
12 the maximum spacing between consecutive medium transitions in
13 order to reduce errors in bit synchronizing to the data. These
14 two run-length constraints on the minimum and maximum spacing
15 between medium transitions are typically referred to as (d,k)
16 respectively.

17 A channel code is typically augmented by an ECC code, such as
18 the well known Reed-Solomon ECC code, which increases the
19 effective SNR by encoding the data according to a minimum Hamming
20 distance which defines the correction power of the ECC code. When
21 noise corrupts a transmitted (recorded) codeword, the received
22 codeword can still be successfully decoded as long as the erroneous
23 bits do not violate the minimum Hamming distance.

24 A typical prior art configuration for the channel and ECC
25 encoders/decoders is illustrated in FIG. 2. The user data 12

1 received from the host are first encoded by an ECC encoder 14 which
2 generates a number of ECC redundancy symbols 16 by dividing the
3 user data represented as a data polynomial by a generator
4 polynomial. The user data 12 are passed through a multiplexer 18
5 and encoded by a channel encoder 20, such as a rate 16/17 RLL
6 encoder 20, and the encoded user data are written to the disk 22
7 by a write modulator 24. The ECC redundancy symbols 16 are then
8 passed through the multiplexer 18, encoded by the channel encoder
9 20, and written to the disk 22 by the write modulator 24. During
10 read back, a data detector 26 detects an estimated data sequence
11 28 from a read signal 30 emanating from a head (not shown)
12 positioned over a selected track of the disk 22. The estimated
13 data sequence 28 is decoded by a channel decoder 32, such as a 16/
14 17 RLL decoder, which implements the inverse operation of the
15 channel encoder 20. The channel decoded data 34 is then decoded
16 by an ECC decoder 36 into user data 38 transmitted to the host.

17 The code rate (input-bits/output-bits) of the channel ENDEC
18 shown in the prior art read/write channel of FIG. 2 is typically
19 low in order to minimize the error propagation for the ECC code.
20 For example, when using a rate 16/17 RLL ENDEC with a byte oriented
21 ECC code, an error in RLL decoding 32 a first byte may propagate
22 into a neighboring byte or bytes which must also be corrected by
23 the ECC code. This error propagation problem increases relative
24 to the number of user data bits encoded by the channel encoder,
25 which directly affects the code rate of the channel ENDEC. Thus,

1 the number of user data bits encoded is typically selected to be
2 low in order to reduce error propagation which places an upper
3 bound on the code rate. This is undesirable since a higher code
4 rate allows more user data to be written to the disk.

5 The code rate limitation of the prior art read/write channel
6 of FIG. 2 is overcome by "commuting" the channel architecture such
7 that the user data is first encoded by the channel encoder, and
8 then encoded by the ECC encoder. This allows for a higher rate
9 channel ENDEC since the error propagation problem is avoided by
10 first ECC decoding the detected data during a read operation, and
11 then passing the ECC corrected data through the channel decoder.

12 An example prior art commuted read/write channel is disclosed
13 in the above referenced U.S. patent application entitled "DISK
14 STORAGE SYSTEM EMPLOYING ERROR DETECTION AND CORRECTION OF CHANNEL
15 CODED DATA, INTERPOLATED TIMING RECOVERY, AND RETROACTIVE/SPLIT-
16 SEGMENT SYMBOL SYNCHRONIZATION." The general configuration and
17 operation of the commuted read/write channel disclosed in the
18 aforementioned patent application is illustrated in FIG. 3. The
19 user data 12 received from the host is first encoded by a high rate
20 channel encoder 40, such as a high rate RLL encoder 40. The
21 channel encoded data 42 passes through a multiplexer 44 and is
22 written to the disk 22 by the write modulator 24. The channel
23 encoded data 42 is simultaneously encoded by an ECC encoder 14
24 which generates the ECC redundancy symbols 16 over the encoded
25 data. The ECC redundancy symbols 16 are then encoded by a low rate

1 channel encoder 46, such as a low rate RLL encoder 46, so that the
2 ECC redundancy symbols 16 satisfy the desired channel constraints
3 when written to the disk 22. The channel encoded ECC redundancy
4 symbols 48 are then passed through multiplexer 44 and written to
5 the disk 22 by the write modulator 24.

6 During a read operation, the data detector 26 detects an
7 estimated data sequence 28 from the read signal 30. The estimated
8 data sequence 28 representing the user data is passed through a
9 multiplexer 50 and input into the ECC decoder 36. The estimated
10 data sequence 28 representing the encoded ECC redundancy symbols
11 is then decoded by a low rate channel decoder 52, such as a low
12 rate RLL decoder 52, which implements the inverse operation of the
13 low rate channel encoder 46. The decoded ECC redundancy symbols
14 54 are then passed through multiplexer 50 and input into the ECC
15 decoder 36 which detects and corrects errors in the estimated data
16 sequence 28 representing the user data. The corrected user data
17 sequence 56 is then decoded by a high rate channel decoder 58, such
18 as a high rate RLL decoder 58, which implements the inverse
19 operation of the high rate channel encoder 40. The decoded user
20 data 38 is then transmitted to the host. The error propagation
21 problem inherent in the prior art read/write channel of FIG. 2 is
22 avoided since the estimated data sequence 28 is first corrected by
23 the ECC decoder 36 and then decoded by the channel decoder 58.

24 Although the prior art commuted read/write channel of FIG. 3
25 is more efficient by allowing for a higher rate channel ENDEC for

1 encoding/decoding the user data, an efficiency problem arises when
2 the number of data bits to be encoded by the high rate channel
3 encoder is not integer divisible by m where the code rate of the
4 channel ENDEC is m/n . Consider, for example, a high rate channel
5 encoder **40** having a code rate of $64/65$. If the number of input
6 bits **12** to be encoded is not integer divisible by 64, then the last
7 part of the data sector must be padded to reach a 64-bit input
8 block. For example, if there are 4128 bits of input data **12**, then
9 the last input block must be padded with

$$\text{mod}(4128/64)=32$$

10
11 pad bits (zero bits) in order to encode the last $64/65$ output block
12 **42**. These encoded pad bits are then written to the disk **22** which
13 is a waste of storage area.

14
15 There is, therefore, a need for a more efficient channel ENDEC
16 in a commuted read/write channel for disk storage systems. In
17 particular, there is a need for a channel ENDEC which reduces or
18 eliminates the number of pad bits when encoding the last part of
19 a data sector.

20 21 SUMMARY OF THE INVENTION

22 The present invention may be regarded as a commuted read/
23 write channel for writing data to and reading data from a disk
24 storage medium. A first channel encoder encodes a first $j-k$ bits
25 of a j -bit data block to generate first encoded data, and an ECC

1 encoder encodes the first encoded data and a remaining k-bits of
2 the data block to generate ECC redundancy symbols comprising a
3 plurality of bits. A second channel encoder encodes the remaining
4 k-bits of the data block and the ECC redundancy symbols to generate
5 second encoded data. The first encoded data and the second encoded
6 data are then output as channel data written to the disk storage
7 medium.

8 The present invention may also be regarded as a method of
9 encoding data in a commuted read/write channel for disk storage
10 systems. The method comprises the steps of first channel encoding
11 a first j-k bits of a j-bit data block to generate first encoded
12 data; ECC encoding the first encoded data and a remaining k-bits
13 of the data block to generate ECC redundancy symbols comprising a
14 plurality of bits; second channel encoding the remaining k-bits of
15 the data block and the ECC redundancy symbols to generate second
16 encoded data; and outputting the first encoded data and the second
17 encoded data as channel data written to a disk storage medium.

18 19 BRIEF DESCRIPTION OF THE DRAWINGS

20 The above and other aspects and advantages of the present
21 invention will be better understood by reading the following
22 detailed description of the invention in conjunction with the
23 drawings, wherein:

24 FIG. 1A shows a typical data format for a magnetic disk
25 storage medium, comprising a plurality of concentric data tracks

1 grouped in predefined zones, where each data track is partitioned
2 into a number of data sectors.

3 FIG. 1B shows a typical format for a data sector.

4 FIG. 2 is a block diagram of a conventional read/write channel
5 wherein the user data is first ECC encoded and then channel encoded
6 (e.g., RLL encoded).

7 FIG. 3 is a block diagram of a prior art commuted read/write
8 channel wherein the user data are first encoded by a high rate
9 channel encoder, then ECC encoded wherein the ECC redundancy
10 symbols are encoded by a low rate channel encoder.

11 FIG. 4 is a block diagram of a commuted read/write channel
12 according to one embodiment of the present invention wherein a
13 first part of a data sector is encoded by a high rate channel
14 encoder, and a last part of the data sector as well as the ECC
15 redundancy symbols are encoded by a low rate channel encoder.

16 FIG. 5A and 5B illustrate an embodiment of the present
17 invention wherein the high rate channel ENDEC is a rate $64/65$
18 channel ENDEC for encoding/decoding a first part of a data sector,
19 and the low rate channel ENDEC is a rate $24/25$ channel ENDEC for
20 encoding/decoding a last part of the data sector and the ECC
21 redundancy symbols.

22 FIG. 6A and 6B illustrate an alternative embodiment of the
23 present invention wherein the high rate channel ENDEC is a $N/(N+1)$
24 channel ENDEC for encoding/decoding a first part of a data sector
25 using two different code rates, and the low rate channel ENDEC is

1 a 24/25 channel ENDEC for encoding/decoding a last part of the data
2 sector and the ECC redundancy symbols.

3 FIG. 7 shows an enumerative encoder implemented using a
4 compare table representing a full enumerative trellis.

5 FIG. 8 shows an enumerative decoder implemented using a
6 compare table representing a full enumerative trellis.

7 FIG. 9 shows an enumerative encoder/decoder (ENDEC) according
8 to an embodiment of the present invention.

9 FIG. 10A and 10B illustrate the encoding operation of the
10 enumerative ENDEC of FIG. 9.

11 FIG. 11A and 11B illustrate the decoding operation of the
12 enumerative ENDEC of FIG. 9.

13 FIG. 12 is a state transition diagram for a rate 64/65 channel
14 code implemented by the enumerative ENDEC of FIG. 9.

15

16

17

18

19

20

21

22

23

24

25

DETAILED DESCRIPTION OF EMBODIMENTS**Commuted Read/Write Channel**

A commuted read/write channel according to one embodiment of the present invention is shown in FIG. 4. During a write operation, the user data **12** received from a host is written to a disk storage medium **22** (magnetic or optical) in predetermined sized data blocks. A first channel encoder **40**, preferably implemented as a high rate RLL encoder **40**, encodes a first $j-k$ bits of a j -bit data block to generate first encoded data **42**, and an ECC encoder **14** encodes the first encoded data **42** and a remaining k -bits of the data block **60** to generate ECC redundancy symbols **16** comprising a plurality of bits. A second channel encoder **46**, preferably implemented as a low rate RLL encoder **46**, encodes the remaining k -bits of the data block and the ECC redundancy symbols **16** to generate second encoded data **48**. The first encoded data **42** and the second encoded data **48** are then output as channel data written to the disk storage medium **22**.

During a read operation, a data detector **26** detects an estimated data sequence **28** from the read signal **30**. The data detector **26** may be implemented using Partial Response (PR) equalization with Maximum Likelihood (ML) sequence detection as is well known in the art. The estimated data sequence **28** representing the first part of a data block encoded by the first channel encoder **40** passes directly through a multiplexer **50** and is input into the ECC decoder **36**. The estimated data sequence **28** representing the

1 last part of the data block 60 together with the encoded ECC
2 redundancy symbols 16 encoded by the second channel encoder 46 are
3 then decoded by a second channel decoder 52, such as a low rate
4 RLL decoder 52, which implements the inverse operation of the
5 second channel encoder 46. The data 54 output by the second
6 channel decoder 52 passes through multiplexer 50 and is input into
7 the ECC decoder 36 which detects and corrects errors in the
8 estimated data sequence 28. The corrected user data sequence 56
9 representing the first part of the data block encoded by the first
10 channel encoder 40 is then decoded by a first channel decoder 58,
11 such as a high rate RLL decoder 58, which implements the inverse
12 operation of the first channel encoder 40. The corrected data
13 sequence 56 representing the last part of the data block 60 which
14 has already been decoded using the second channel decoder 52 passes
15 directly through multiplexer 59. The decoded user data 38 output
16 by multiplexer 59 is then transmitted to the host.

17 A more specific embodiment of the present invention is
18 illustrated in FIG. 5A and 5B. In this embodiment, a data block
19 to be written to the disk storage medium 22 is a data sector
20 comprising 512 bytes of user data and 4 CRC check bytes for a total
21 of 516 bytes or 4128 bits of input data 12. Also in this
22 embodiment, the ECC encoder 14 generates 40 ECC redundancy symbols
23 16. The format of the data encoded by the channel encoders 40 and
24 46 is illustrated in FIG. 5B, with the corresponding circuitry
25 illustrated in FIG. 5A. The first channel encoder 40 of FIG. 5A

1 is implemented with a code rate of 64/65 and the first 4096 bits
2 (integer divisible by 64) of input data 12 are encoded by the first
3 channel encoder 40 to generate 4160 bits of first encoded data 42.
4 Bits 4096-4127 (32 bits) 60 of the input data 12 pass through
5 multiplexer 62 and are encoded by the second channel encoder 46.
6 The 4160 bits of first encoded data 42 pass through multiplexer 64
7 and are input into the ECC encoder 14, and then the last 32 bits
8 of input data 60 pass through multiplexer 64 and are input into
9 the ECC encoder 14. The 320 bits of ECC redundancy symbols 16 pass
10 through multiplexer 62 and are encoded by the second channel
11 encoder 46. The first encoded data 42 passes through multiplexer
12 66 and is written to the disk storage medium 22 by a write
13 modulator 24, followed by the second encoded data 48 passing
14 through multiplexer 66 and written to the disk storage medium 22.

15 In the embodiment of FIG. 5A, the code rate of the second
16 channel encoder 46 is 24/25. Because 24 does not integer divide
17 into 352 (the last 32 bits 60 of the input data 12 plus the 320
18 bits of ECC redundancy symbols 16), 8 pad bits are added to the
19 last input block of the second channel encoder 46 so that the last
20 input block comprises 24 bits. That is, if the code rate of the
21 second encoder 46 is m/n , then pad bits are added such that the
22 number of bits in the last part of the data sector 60 plus the ECC
23 redundancy symbols is integer divisible by m .

24 When reading a data sector from the disk storage medium 22,
25 the first part of the data sector which was encoded by the 64/65

1 channel encoder 40 passes through multiplexer 50 and is input into
2 an ECC decoder 36. The last part of the data sector and the ECC
3 redundancy symbols 16 are decoded by a 24/25 channel decoder 52
4 which implements the inverse operation of the 24/25 channel
5 encoder 46. The rate 24/25 decoded data 54 passes through
6 multiplexer 50 and are input into the ECC decoder 36. Bits 0-4095
7 of the data sector are decoded by a 64/65 channel decoder 58 which
8 implements the inverse operation of the 64/65 channel encoder 40.
9 Bits 4096-4127 of the data sector, which have already been decoded
10 by the 24/25 channel decoder 52, pass directly through multiplexer
11 59. The decoded user data 38 output by multiplexer 59 is then
12 transmitted to the host.

13 In an alternative embodiment of the present invention,
14 illustrated in FIG. 6A and 6B, the first channel encoder 40 is
15 programmable in that the code rate $N/(N+1)$ can be adjusted for
16 different values of N. This embodiment is more efficient because
17 it eliminates the need for pad bits added to the last input block
18 for the second encoder 46 (e.g., the 8 pad bits in FIG. 5B). The
19 first encoder 40 is programmed to a code rate of 64/65 to encode
20 bits 0-4095 (4096 is integer divisible by 64), and then it
21 is programmed to a code rate of 16/17 to encode bits 4096-4111
22 (4112-4096=16 is integer divisible by 16). Bits 4112-4127 (16
23 bits) 60 of the input data 12 pass through multiplexer 62 and are
24 encoded by the second channel encoder 46, followed by the 320 bits
25 of ECC redundancy symbols. Because 24 integer divides into 336

1 (the last 16 bits 60 of the input data 12 plus the 320 bits of ECC
2 redundancy symbols 16), pad bits are not added to the last input
3 block of the 24/25 channel encoder 46 as in the embodiment of FIG.
4 5A.

5 When reading a data sector from the disk the commuted read/
6 write channel of FIG. 6A operates essentially the same as that of
7 FIG. 5A except that the $N/(N+1)$ channel decoder 58 is programmed
8 to decode bits 0-4096 at a code rate of 64/65 and bits 4096-4111
9 at a code rate of 16/17 to implement the inverse operation of the
10 $N/(N+1)$ channel encoder 40. Implementation details concerning a
11 suitable embodiment for the programmable $N/(N+1)$ channel encoder
12 40 and $N/(N+1)$ channel decoder 58 of FIG. 6A are disclosed in the
13 above referenced copending patent application entitled "A COST-
14 EFFECTIVE HIGH-THROUGHPUT ENUMERATIVE ENDEC EMPLOYING A PLURALITY
15 OF SEGMENTED COMPARE TABLES," the relevant sections of which are
16 set forth below.

17 18 Enumerative Encoder - Full Compare Table

19 An overview of an enumerative encoder comprising a full
20 compare table 70 of an enumerative trellis is shown in FIG. 7. To
21 initialize the encoder, the input dataword 72 represented as an
22 integer is loaded into register 74 through multiplexer 76, and an
23 initial state 78 is loaded into register 80 through multiplexer
24 82. The output of register 74 is input into a D-C computation
25 block 84, and the output of register 80 (the current state 86) is

1 input into a state table 88.

2 At each branch in the enumerative trellis, the D-C
3 computation block 84 computes the value D-C where D is the input
4 data and C is the current state value which is provided by a
5 compare table 70 over line 90. The compare table 70 generates the
6 current state value C based on the current bit position 92 of the
7 input data and the current state 86 of the enumerative trellis.
8 The D-C computation block 84 outputs the subtraction value D-C 94
9 and a sign bit 96. The subtraction value D-C 94 is loaded back
10 into register 74 through multiplexer 76 if the sign bit 96 is zero
11 in order to update the input value D (i.e., if D is greater than
12 C update the input value); otherwise, the input value D is
13 unmodified.

14 The sign bit 96 is input into the state table 88 which
15 responds by selecting the next state in the enumerative trellis.
16 That is, if D-C is positive then the state table 88 selects the
17 corresponding branch to the next state, otherwise the state table
18 88 selects the branch corresponding to a negative value. The
19 selected state is output over line 98 and loaded into register 80
20 through multiplexer 82 so that it becomes the current state 86 for
21 encoding the next input bit.

22 The sign bit 96 output by the D-C computation block 84 is
23 inverted by inverter 100. If there is only one branch leaving the
24 current state of the enumerative trellis, then a force circuit 102
25 forces the output bit 104 of the encoder to be a "0" or a "1";

1 otherwise the output of the inverter 100 is selected as the encoder
2 output 104 through multiplexer 106.

4 Enumerative Decoder - Full Compare Table

5 An overview of an enumerative decoder comprising a full
6 compare table 108 of an enumerative trellis is shown in FIG. 8.
7 To initialize the decoder, an initial state 110 is loaded into
8 register 112 through multiplexer 114. The output 116 of register
9 112, which represents the current state, is input into a state
10 table 118. The state table 118 in the decoder of FIG. 8 is the
11 same as the state table 88 in the encoder of FIG. 7; the difference
12 being that the decoder uses the current state 116 to construct an
13 integer output value (dataword) based on the input codeword 120.

14 As each bit of the input codeword 120 is processed, the
15 compare table 108 outputs a current state value 121 based on the
16 current state 116 and the current bit position 122 of the input
17 codeword 120. The state value 121 is applied to an AND gate 124;
18 if the current bit of the input codeword 120 is a "1" bit, then
19 the state value 121 is accumulated by accumulator 126, otherwise
20 the state value 121 is ignored. The state table 118 updates the
21 next state (via multiplexer 114 and register 112) based on the
22 current input bit of the input codeword 120; that is, the branch
23 corresponding to a "1" or "0" bit is selected. If there is only
24 one branch leaving the current state, then the input bit of the
25 input codeword 120 is ignored. Once the entire input codeword 120

has been processed, the entire enumerative trellis will have been traversed and the decoded dataword **128** is output by the accumulator **126**.

Enumerative ENDEC - Segmented Compare Table

Encoding or decoding data on-the-fly at the channel rate using the enumerative ENDEC disclosed in FIG. 7 and FIG. 8 would require a k-bit subtraction (or addition) on every channel clock, where k is the number of bits in an input dataword. For example, a rate 64/65 enumerative ENDEC would require a 64-bit subtraction on every channel clock when encoding an input dataword. This is not practical for high speed communication channels, such as in disk storage systems.

One solution to this problem is to duplicate the ENDEC circuitry and process multiple codewords at a time. This reduces the clock rate of the ENDEC by CLK/n , where CLK is the channel clock and n is the number of duplicated ENDECs. However, duplicating the ENDEC circuitry is not cost effective, particularly when designing high rate codes.

FIG. 9 shows an embodiment of the present invention which processes multiple codewords simultaneously without duplicating the ENDEC circuitry in its entirety. Instead, the full compare tables (**70** and **108**) of FIG. 7 and FIG. 8 are segmented into a plurality of segmented compare tables each of which represents a segment of the enumerative trellis and each of which operate on a

1 corresponding segment of an input dataword.

2 The example embodiment of FIG. 9 implements a rate 64/65
3 enumerative ENDEC where the full compare table representing the
4 full enumerative trellis has been sliced into eight segmented
5 compare tables. Each segmented compare table is part of a slice
6 circuit 130_0 - 130_7 where each slice circuit 130_0 - 130_7 comprises the
7 same circuitry shown in FIG. 7 and FIG. 8 with the full compare
8 table (70 and 108) replaced by a segmented compare table.

9 During encoding, the first seven slice circuits 130_0 - 130_6
10 generate 56 bits of an output codeword (8 bits each) and the last
11 slice circuit 130_7 generates the last 9 bits of an output codeword.
12 During decoding, the first seven slice circuits 130_0 - 130_6 process
13 56 bits of an input codeword (8 bits each) and the last slice
14 circuit 130_7 processes the last 9 bits of an input codeword.
15 During both encoding and decoding, the eight slice circuits 130_0 -
16 130_7 operate on eight codewords simultaneously as explained in
17 further detail below with reference to FIGs. 10A-11B. Thus, the
18 clock rate of the enumerative ENDEC can operate at 1/8 the rate of
19 the channel clock.
20

21 The buffer IO_BUF 132 in FIG. 9 stores the incoming data bits
22 during the encoding process and the outgoing data bits during the
23 decoding process. During encoding, on each ENDEC clock the IO_BUF
24 132 receives one byte (8 bits) of input data, and during decoding
25 it outputs one byte of decoded output. Because the code rate is

1 64/65, after receiving 64 bytes during the encoding process there
2 will be an extra byte ready to output. Thus, the input to IO_BUF
3 132 must pause for one ENDEC clock to allow this byte to be output.
4 Similarly, during the decoding process there is only 64 bytes
5 output for every 65 bytes recieved; therefore, the decoding output
6 will stop for one ENDEC clock after every 64th output byte.

7 A plurality of shift registers SR0-SR8 134₀-134₈ are provided
8 for storing the bits of the encoded output codewords during the
9 encoding process, and for storing the bits of the encoded input
10 codewords during a decoding process. During the encoding process,
11 the first eight registers 134₀-134₇ output the first 64 bits of the
12 encoded codeword (8 bits from each register), and the last register
13 134₈ outputs the 65th bit of the encoded codeword. After an
14 initial delay, the codeword data begins to be transferred from the
15 shift registers SR0-SR8 134₀-134₈ to the channel. During
16 subsequent ENDEC clocks, one byte of the encoded codeword is
17 transferred from the appropriate shift register SR0-SR8 134₀-134₈
18 to the channel. During the decoding process, on each ENDEC clock
19 a byte of an input codeword is loaded into the corresponding
20 register 134₀-134₇ and the 65th bit is loaded into register 134₈.
21 After an initial delay, the last slice circuit 130₇ outputs a 64-
22 bit decoded output dataword on every 8th ENDEC clock which is
23 transmitted over line 140 and stored in IO_BUF 132.

24 The size of the shift registers 134₀-134₈ varies for each
25

1 slice circuit 130_0-130_7 , and the amount of memory used in each
2 register 134_0-134_8 varies depending on whether the ENDEC is
3 encoding or decoding. During encoding, the entire storage area is
4 utilized in the first four registers 134_0-134_3 and a decreasing
5 amount of storage area is used in the next four registers 134_4-
6 134_7 . During decoding, the situation is reversed as illustrated
7 in FIG. 9. The size of the shift registers 134_0-134_8 varies due
8 to the overlapping method in which the codewords are encoded and
9 decoded as will now be described.

10 During the encoding process, the IO_BUF 132 first receives
11 eight bytes of input data during the first eight ENDEC clocks to
12 form a 64-bit input dataword. The 64-bit input dataword is then
13 transferred to the first slice circuit 130_0 over line 136_0 . During
14 the next eight ENDEC clocks, the first slice circuit 130_0 computes
15 bits 64-57 of the output codeword which are shifted serially into
16 shift register SR0 134_0 . Concurrently, the next eight input data
17 bytes are input into IO_BUF 132 such that the next 64-bit input
18 dataword is ready to be processed. On the next clock cycle, the
19 first slice circuit 130_0 transfers a next value NXT_VAL and a next
20 state NXT_STATE to the second slice 130_1 over line 136_1 . The
21 NXT_VAL represents the current state value input into the D-C
22 computation block (conceptually, the value stored in register 74
23 of FIG. 7). The NXT_STATE represents the current state in the
24 enumerative trellis after processing the first eight bits
25

1 (conceptually, the value stored in register 80 of FIG. 7).
2 Concurrently, the next 64-bit input dataword is transferred from
3 the IO_BUF 132 to the first slice circuit 130₀ over line 136₀.
4 During the next eight ENDEC clock cycles, the following operations
5 are performed concurrently:

- 6 1. the next eight input data bytes are input into IO_BUF 132 such
7 that the next 64-bit input dataword is ready to be processed;
- 8 2. the first slice circuit 130₀ computes bits 64-57 of the next
9 output codeword which are shifted serially into shift
10 register SR0 134₀; and
- 11 3. the second slice circuit 130₁ computes bits 56-49 of the first
12 output codeword which are shifted serially into shift
13 register SR1 134₁.

14 On the eighth ENDEC clock, the NXT_VAL and NXT_STATE values are
15 shifted from the current slice circuits 130₀-130₆ to the following
16 slice circuits 130₁-130₇ and the above process is reiterated. This
17 process continues for 64 ENDEC clocks after which the output
18 registers SR0-SR8 134₀-134₈ receive the entire 65-bit output
19 codeword for the first codeword, as well as portions of the
20 following codewords. Note that on the 64th ENDEC clock, the last
21 two bits (bits 0 and 1) of the 65-bit output codeword are
22 generated. After 57 bits have been shifted into SR0 134₀,
23 outputting to the channel begins. At this point, the first byte
24 of the first codeword is stored in SR0(49:56), the second byte is
25

1 stored in SR1(41:48), the seventh byte is stored in SR6(1:8), and
2 the first bit of the last byte is stored in SR7(0). Registers SR0-
3 SR8 134_0 - 134_8 continue to shift during outputting to the channel
4 so that the output from SR1 134_1 is read from SR1(42:49) since it
5 is read one ENDEC clock after the byte from SR0 134_0 is read. The
6 output from SR6 134_6 is read from SR6(7:14) since it is taken six
7 ENDEC clocks after SR0 134_0 is read. As explained above, on every
8 64th ENDEC clock the input into IO_BUF 132 is paused for one ENDEC
9 clock to allow the extra encoded byte to be transferred from the
10 shift registers SR0-SR8 134_0 - 134_8 to the channel.

11 The encoding process is further understood with reference to
12 FIG. 10A and 10B which illustrate the encoding operation for two
13 consecutive input datawords. In FIG. 10A, the first input dataword
14 has been processed by the first slice circuit 130_0 for eight ENDEC
15 clocks to generate the first byte of the first output codeword
16 (denoted CW_{00}) which has been shifted into shift register SR0 134_0 .
17 On the eighth ENDEC clock, the NXT_VAL and NXT_STATE values are
18 transferred to the second slice circuit 130_1 concurrent with the
19 next input codeword being loaded into the first slice circuit 130_0 .
20 As illustrated in FIG. 10B, during the next eight ENDEC clocks the
21 first slice circuit 130_0 computes the first byte of the next output
22 codeword (denoted CW_{10}) which is shifted into shift register SR0
23 134_0 concurrent with the second slice circuit 130_1 computing the
24 second byte of the first output codeword (denoted CW_{01}) which is
25

1 shifted into shift register SR1 134₁. The slice circuits 130₀-130₆
2 then transfer the NXT_VAL and NXT_STATE values to the following
3 slice circuits 130₁-130₇, and the process continues.

4 During the decoding process, the data bits received from the
5 channel are stored in the shift registers SR0(0:7)-SR8(0:7)
6 starting with SR0(0:7). The first byte of a received codeword is
7 stored in SR0(0:7) on the first ENDEC clock, the next byte is
8 stored in SR1(0:7) on the next ENDEC clock, and so on. After the
9 first byte of a received codeword is loaded into register SR0(0:7)
10 then during the next eight ENDEC clock cycles the bits are shifted
11 serially over line 138₀ from SR0(7) into the first slice circuit
12 130₀ for decoding (concurrent with loading shift registers
13 SR1(0:7)-SR8(0:7) 134₁-134₈ with the subsequent bytes of the
14 received codeword). On the eighth ENDEC clock, the result of
15 decoding the first byte of the received codeword (the accumulated
16 value NXT_VAL and the ending state NXT_STATE) are transferred to
17 the next slice circuit 130₁ over line 136₁. Conceptually, the
18 NXT_VAL corresponds to the value accumulated in accumulator 126 of
19 FIG. 8, and the NXT_STATE corresponds to the value stored in
20 register 112 of FIG. 8. During the next eight ENDEC clock cycles,
21 the following operations are performed concurrently:

- 22 1. the next eight bytes for the next received codeword are input
23 into the shift registers SR0(0:7)-SR8(0:7) 134₀-134₈;
- 24 2. the first slice circuit 130₀ decodes the first byte of the
25

1 next codeword; and

2 3. the second slice circuit **130₁** decodes the second byte of the
3 first received codeword using the **NXT_VAL** and **NXT_STATE**
4 values computed by the first slice circuit **130₀** for the first
5 byte and transferred over line **136₁**.

6 On the next ENDEC clock, the **NXT_VAL** and **NXT_STATE** values are
7 shifted from the current slice circuits **130₀-130₆** to the following
8 slice circuits **130₁-130₇** and the above process is reiterated. This
9 process continues for 64 ENDEC clocks after which the last slice
10 circuit **130₇** contains the accumulated value for the integer
11 representing the first decoded dataword; this value **CUR_VAL** is
12 transferred over line **140** to the **IO_BUF 132**. On every 8th ENDEC
13 clock thereafter, the last slice circuit **130₇** outputs the integer
14 value **CUR_VAL** for a next decoded dataword.

15 The decoding process is further understood with reference to
16 FIG. 7A and 7B which illustrate the decoding operation for two
17 consecutive received codewords. In FIG. 7A, the first byte of a
18 received codeword (denoted **CW₀₀**) has been stored in register
19 **SR0(0:7) 134₀** on the first ENDEC clock and the second byte (denoted
20 **CW₀₁**) has been stored in register **SR1(0:7) 134₁** on the second ENDEC
21 clock. Also on the second ENDEC clock, the first slice circuit
22 **130₀** processes the first bit of the first byte **CW₀₀** from **SR0(7)** to
23 begin the decoding process. As shown in FIG. 7B, after eight ENDEC
24 clocks the first slice circuit **130₀** has finished decoding the first
25

1 byte CW_{00} of the received codeword and transfers over line 136_1 the
2 NXT_VAL and the NXT_STATE values to the next slice circuit 130_1 .
3 Concurrently, the first byte of the next codeword (denoted CW_{10})
4 is loaded into register $SR0(0:7)$ 134_0 . On the next ENDEC clock
5 the second byte of the next codeword (denoted CW_{11}) is loaded into
6 register $SR1(0:7)$ 134_1 and the second slice circuit 130_1 processes
7 the first bit of the second byte CW_{01} in the first codeword from
8 $SR1(14)$. After eight ENDEC clocks the slice circuits 130_0-130_6
9 transfer the NXT_VAL and NXT_STATE values to the following slice
10 circuits 130_1-130_7 and the process continues.

12 The following is a glossary of terms used in FIG. 9:

13 $BDI(7:0)$ - Incoming data from the buffer encoding. This data is
14 byte transferred to IO_BUF 132 to construct the 64-bit
15 input codewords.

16 $BLK_IN(63:0)$ - A 64-bit input dataword transferred from IO_BUF 132
17 to the first slice circuit 130_0 during the encoding
18 operation.

19 $SR0-SR7$ - The shift registers 134_0-134_7 of varying length ($SR0=57$
20 bits; $SR1=50$ bits; $SR2=43$ bits; $SR3=36$ bits; $SR4=36$
21 bits; $SR5=43$ bits; $SR6=50$ bits; $SR7=57$ bits). Each shift
22 register has a parallel load into bits $(0:7)$ for
23 inputting data for decoding, and an 8-bit parallel
24 output from varying points within the registers for
25 offloading the encoded data at the appropriate time.

1 SR8 - The 8-bit shift register 134_8 which accumulates the 65th bit
 2 during the encoding operation, and which stores the 65th
 3 bit during the decoding operation.

4 DEC_BITS(7:0) - Data bytes received from the channel during a read
 5 operation which are stored in the shift registers SR0-
 6 SR7(0:7) 134_0-134_7 .

7 DEC_LST_BIT - The 65th bit of a received codeword during decoding
 8 which is stored in shift register SR8 134_8 .

9 ENC*_BITS(7:0) - The encoded data bytes transferred from the shift
 10 registers SR0-SR7 134_0-134_7 to the channel during a write
 11 operation.

12 BLK_OUT(63:0) - A 64-bit decoded dataword output by the last slice
 13 circuit 130_7 during the decoding operation.

14 BDO(7:0) - The decoded data output from IO_BUF 132 a byte at a
 15 time.

16 ENC_8_0 through ENC_64_57 - The eight slice circuits 130_0-130_7
 17 where the numerals refer to the bit positions of an input
 18 dataword encoded by each slice.

20 Rate 64/65 State Transition Table

21 A state transition diagram for a rate 64/65 code is shown in
 22 FIG. 12. A corresponding enumerative trellis can be constructed
 23 from this state transition diagram using well known techniques.
 24 The segmented compare tables in the slice circuits 130_0-130_7 of
 25

FIG. 9 are then constructed by slicing the enumerative trellis into eight segments.

Rate	$N/(N+1)$	Enumerative	ENDEC
------	-----------	-------------	-------

The example embodiment of the enumerative ENDEC illustrated in FIG. 9 implements a rate 64/65 code. However, the circuitry can be modified to accomodate any code rate and the enumerative trellis can be segmented arbitrarily rather than on byte boundaries. Furthermore, the circuitry of FIG. 9 could be modified to shorten the code rate on-the-fly to accomodate different code rates by bypassing a selected number of the encoding/decoding sections. For the implementation shown in FIG. 9, it is convenient to shorten the code rate by a multiple of 8. For example, to shorten the ENDEC to a code rate of 56/57, the input data to be encoded and the received data to be decoded would be sent directly to the slice circuit ENC_56_49 130₁. Alternatively, the input data and received data could be sent directly to the slice circuit ENC_8_0 130₇, which would implement a rate 8/9 code. Thus, the enumerative ENDEC illustrated in FIG. 9 may be implemented as a rate $N/(N+1)$ ENDEC in order to implement the rate $N/(N+1)$ channel encoder 40 and the rate $N/(N+1)$ channel decoder 58 of FIG. 6A.

The objects of the invention have been fully realized through the embodiments disclosed herein. Those skilled in the art will appreciate that the various aspects of the invention can be

1 achieved through different embodiments without departing from the
2 essential function. For example, code rates other than the
3 disclosed 64/65, 24/25 and 16/17 could be employed. The disclosed
4 embodiments are thus illustrative and not intended to limit the
5 scope of the present invention as appropriately construed from the
6 following claims.

Claims:

1. A commuted read/write channel for writing data to and reading data from a disk storage medium, comprising:
 - (a) a first channel encoder for encoding a first j - k bits of a j -bit data block to generate first encoded data;
 - (b) an ECC encoder for encoding the first encoded data and a remaining k -bits of the data block to generate ECC redundancy symbols comprising a plurality of bits;
 - (c) a second channel encoder for encoding the remaining k -bits of the data block and the ECC redundancy symbols to generate second encoded data; and
 - (d) an output for outputting the first encoded data and the second encoded data as channel data written to the disk storage medium.
2. The commuted read/write channel as recited in claim 1, wherein a code rate of the first channel encoder is greater than a code rate of the second channel encoder.
3. The commuted read/write channel as recited in claim 1, wherein the first channel encoder is an enumerative encoder.
4. The commuted read/write channel as recited in claim 3, wherein the enumerative encoder has a programmable code rate.

1 5. The commuted read/write channel as recited in claim 4,
2 wherein:

3 (a) the first channel encoder encodes a first part of the j-
4 k bits of the data block according to a first code rate
5 of m_0/n_0 ; and

6 (b) the first channel encoder encodes a last part of the j-
7 k bits of the data block according to a second code rate
8 of m_1/n_1 .

9
10 6. The commuted read/write channel as recited in claim 5,
11 wherein:

12 (a) the first part of the j-k bits of the data block is
13 integer divisible by m_0 ; and

14 (b) the last part of the j-k bits of the data block is
15 integer divisible by the m_1 .

16
17 7. The commuted read/write channel as recited in claim 5,
18 wherein:

19 (a) the second channel encoder encodes m_2 bits of input data
20 into n_2 bits of output data; and

21 (b) the bits of the ECC redundancy symbols plus the k-bits
22 of the data block is integer divisible by m_2 .

23
24 8. A method of encoding data in a commuted read/write channel for
25

1 disk storage systems, comprising the steps of:

- 2 (a) first channel encoding a first $j-k$ bits of a j -bit data
3 block to generate first encoded data;
4 (b) ECC encoding the first encoded data and a remaining k -
5 bits of the data block to generate ECC redundancy symbols
6 comprising a plurality of bits;
7 (c) second channel encoding the remaining k -bits of the data
8 block and the ECC redundancy symbols to generate second
9 encoded data; and
10 (d) outputting the first encoded data and the second encoded
11 data as channel data written to a disk storage medium.

12
13 9. The method as recited in claim 8, wherein a code rate of the
14 first channel encoding step is greater than a code rate of
15 the second channel encoding step.

16
17 10. The method as recited in claim 8, wherein the first channel
18 encoding step comprises the step of enumerative encoding.

19
20 11. The method as recited in claim 10, wherein the step of
21 enumerative encoding comprises the step of programming a code
22 rate.

23
24 12. The method as recited in claim 11, wherein the first channel
25 encoding step comprises the steps of:

- 1 (a) encoding a first part of the $j-k$ bits of the data block
2 according to a first code rate of m_0/n_0 ; and
3 (b) encoding a last part of the $j-k$ bits of the data block
4 according to a second code rate of m_1/n_1 .

5
6 13. The method as recited in claim 12, wherein:

- 7 (a) the first part of the $j-k$ bits of the data block is
8 integer divisible by m_0 ; and
9 (b) the last part of the $j-k$ bits of the data block is
10 integer divisible by the m_1 .

11
12 14. The method as recited in claim 12, wherein:

- 13 (a) the second channel encoding step comprises the step of
14 encoding m_2 bits of input data into n_2 bits of output
15 data; and
16 (b) the bits of the ECC redundancy symbols plus the k -bits
17 of the data block is integer divisible by m_2 .

18
19 15. A commuted read/write channel for transmitting data through a
20 communication medium, comprising:

- 21 (a) a first channel encoder for encoding a first $j-k$ bits of
22 a j -bit data block to generate first encoded data;
23 (b) an ECC encoder for encoding the first encoded data and
24 a remaining k -bits of the data block to generate ECC
25

- 1 redundancy symbols comprising a plurality of bits;
2 (c) a second channel encoder for encoding the remaining k-
3 bits of the data block and the ECC redundancy symbols to
4 generate second encoded data; and
5 (d) an output for outputting the first encoded data and the
6 second encoded data as channel data transmitted through
7 the communication medium.

- 8
9 16. The commuted read/write channel as recited in claim 15,
10 wherein a code rate of the first channel encoder is greater
11 than a code rate of the second channel encoder.
12
13 17. The commuted read/write channel as recited in claim 15,
14 wherein the first channel encoder is an enumerative encoder.
15
16 18. The commuted read/write channel as recited in claim 17,
17 wherein the enumerative encoder has a programmable code rate.
18
19 19. The commuted read/write channel as recited in claim 18,
20 wherein
21 (a) the first channel encoder encodes a first part of the j-
22 k bits of the data block according to a first code rate
23 of m_0/n_0 ; and
24 (b) the first channel encoder encodes a last part of the j-
25 k bits of the data block according to a second code rate

1 of m_1/n_1 .

2
3 20. The commuted read/write channel as recited in claim 19,
4 wherein:

5 (a) the first part of the $j-k$ bits of the data block is
6 integer divisible by m_0 ; and

7 (b) the last part of the $j-k$ bits of the data block is
8 integer divisible by m_1 .

9
10 21. The commuted read/write channel as recited in claim 19,
11 wherein:

12 (a) the second channel encoder encodes m_2 bits of input data
13 into n_2 bits of output data; and

14 (b) the bits of the ECC redundancy symbols plus the k -bits
15 of the data block is integer divisible by m_2 .

16
17 22. A method of encoding data in a commuted read/write channel for
18 transmitting data through a communication medium, comprising
19 the steps of:

20 (a) first channel encoding a first $j-k$ bits of a j -bit data
21 block to generate first encoded data;

22 (b) ECC encoding the first encoded data and a remaining k -
23 bits of the data block to generate ECC redundancy symbols
24 comprising a plurality of bits;
25

(c) second channel encoding the remaining k-bits of the data block and the ECC redundancy symbols to generate second encoded data; and

(d) outputting the first encoded data and the second encoded data as channel data transmitted through the communication medium.

23. The method as recited in claim 22, wherein a code rate of the first channel encoding step is greater than a code rate of the second channel encoding step.

24. The method as recited in claim 22, wherein the first channel encoding step comprises the step of enumerative encoding.

25. The method as recited in claim 24, wherein the step of enumerative encoding comprises the step of programming a code rate.

26. The method as recited in claim 25, wherein the first channel encoding step comprises the steps of:

(a) encoding a first part of the j-k bits of the data block according to a first code rate of m_0/n_0 ; and

(b) encoding a last part of the j-k bits of the data block according to a second code rate of m_1/n_1 .

1 27. The method as recited in claim 26, wherein:

2 (a) the first part of the $j-k$ bits of the data block is
3 integer divisible by m_0 ; and

4 (b) the last part of the $j-k$ bits of the data block is
5 integer divisible by the m_1 .

6
7 28. The method as recited in claim 26, wherein:

8 (a) the second channel encoding step comprises the step of
9 encoding m_2 bits of input data into n_2 bits of output
10 data; and

11 (b) the bits of the ECC redundancy symbols plus the k -bits
12 of the data block is integer divisible by m_2 .

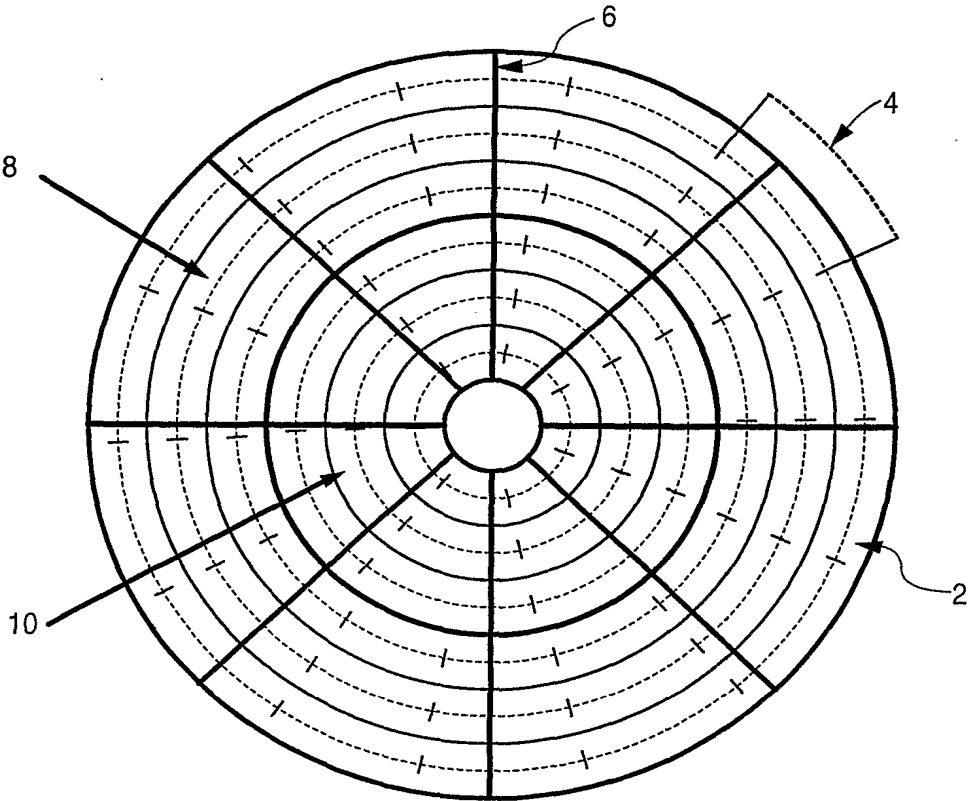


FIG. 1A
(Prior Art)

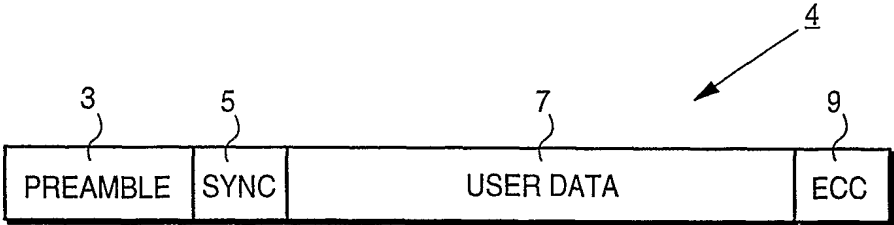


FIG. 1B
(Prior Art)

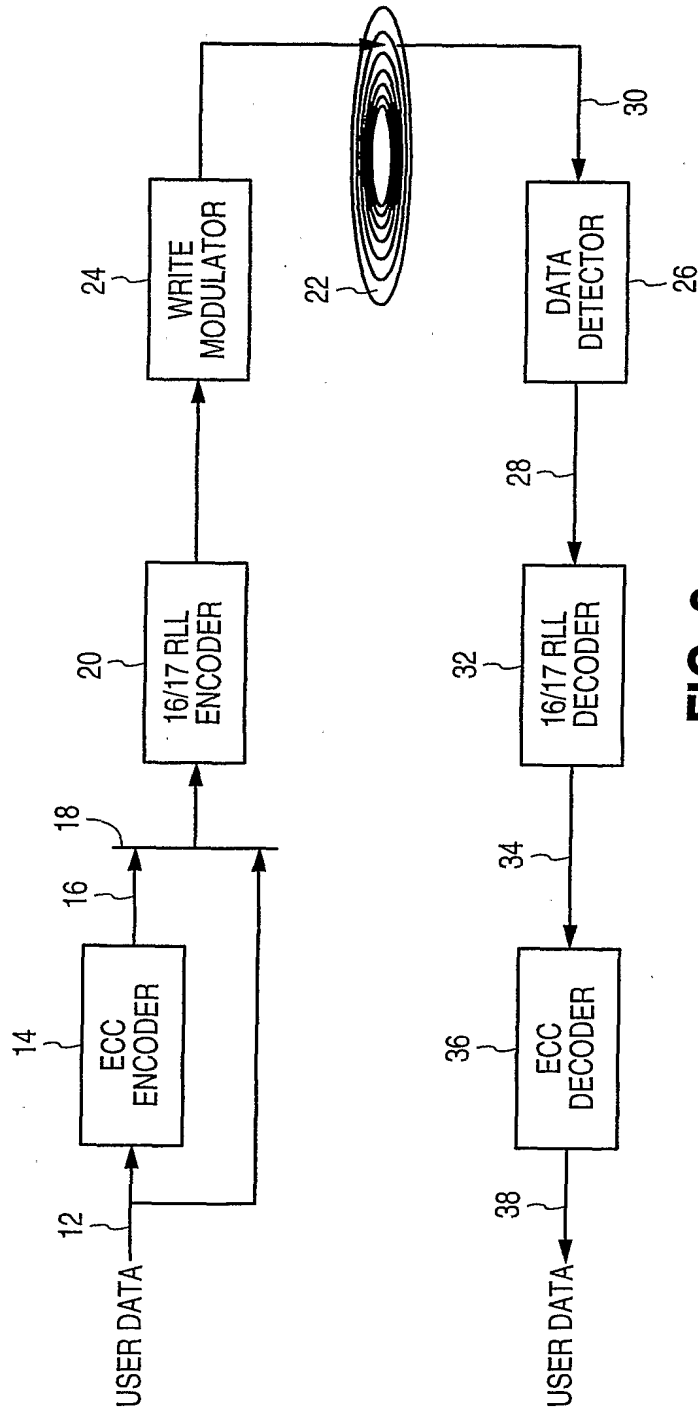


FIG. 2
(Prior Art)

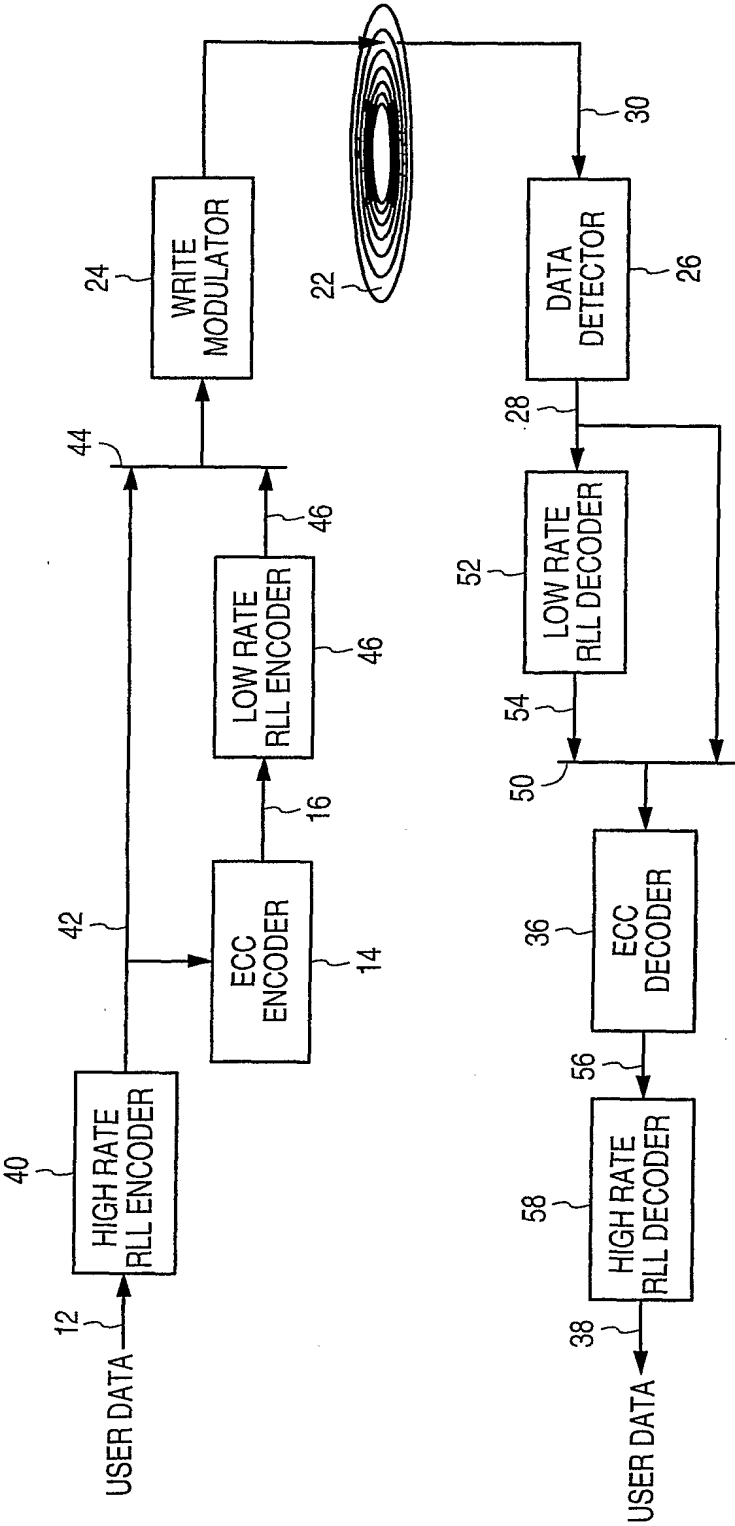


FIG. 3
(Prior Art)

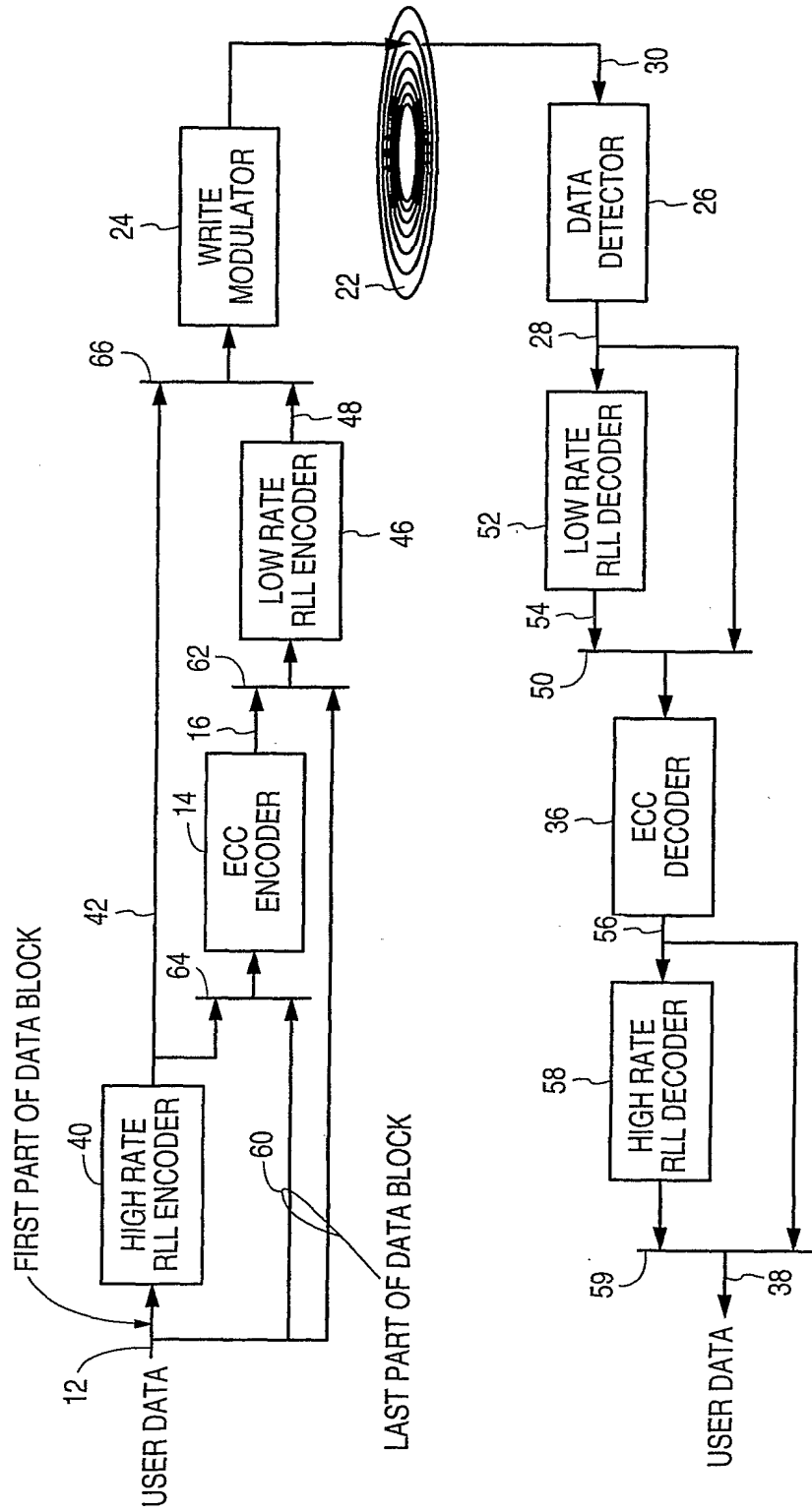


FIG. 4

FIG. 5A

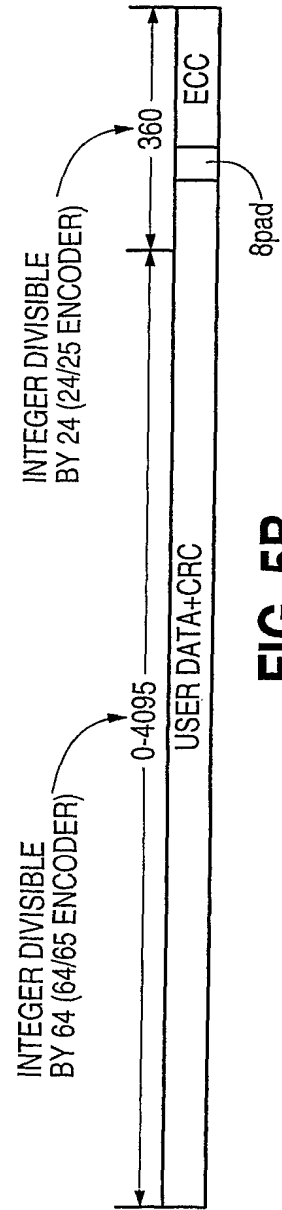
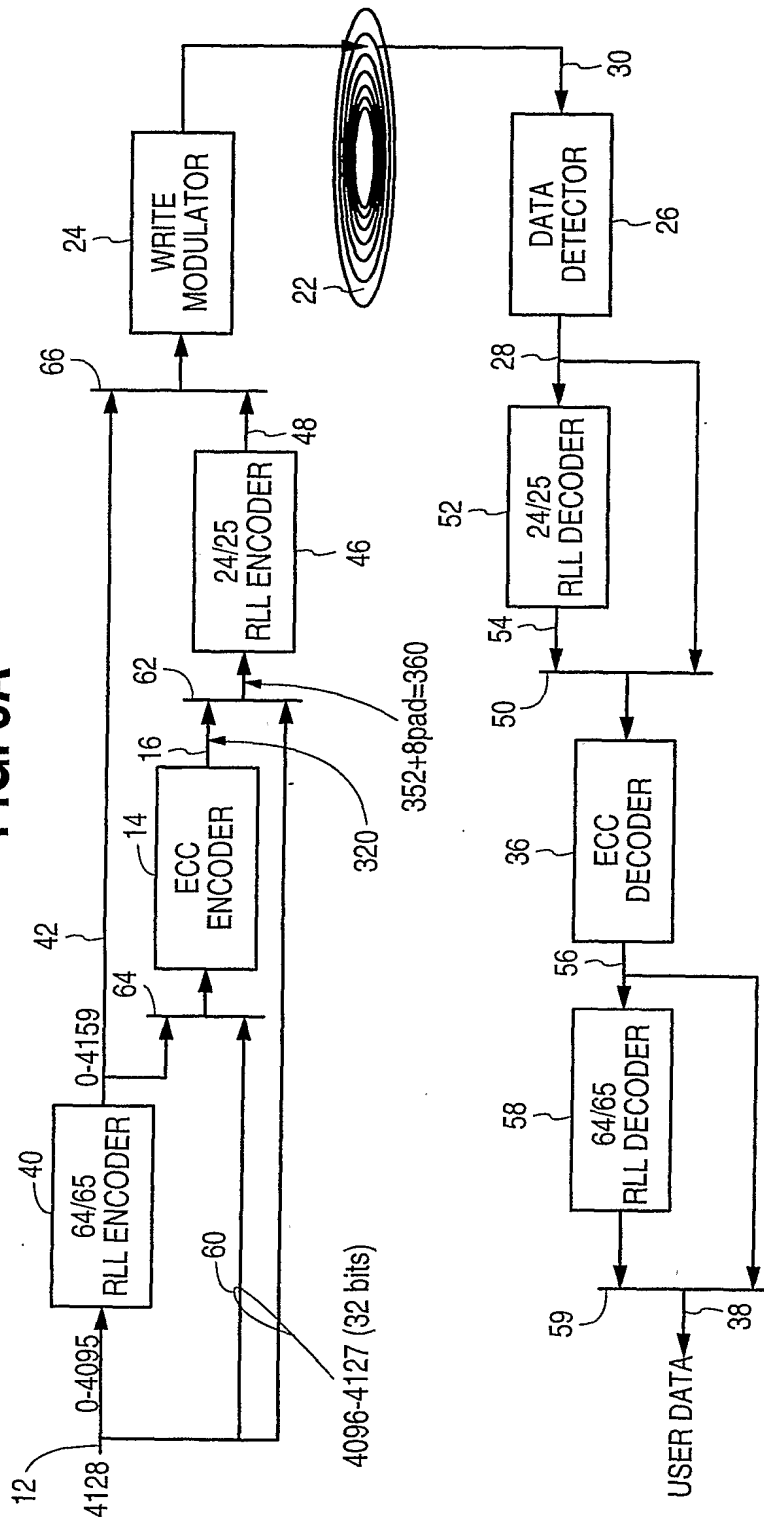
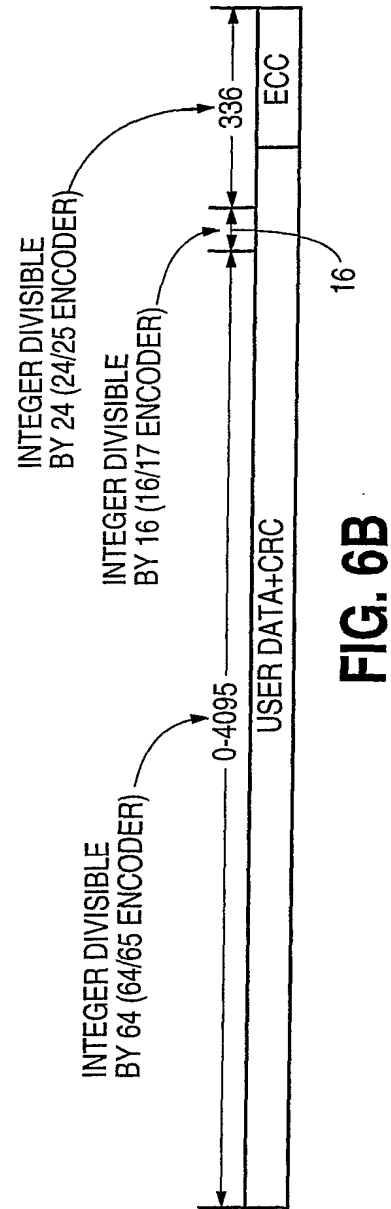
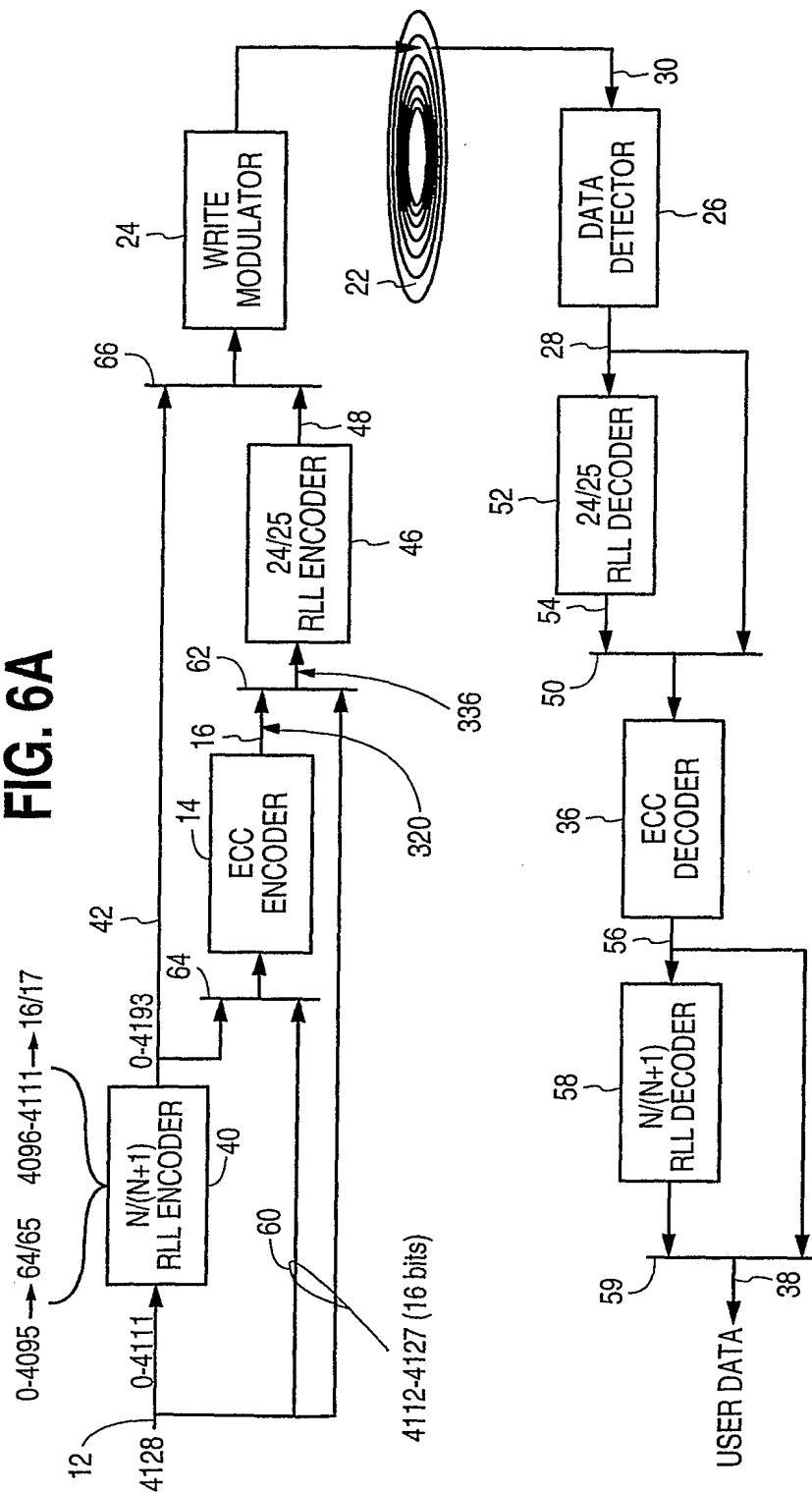


FIG. 5B

FIG. 6A



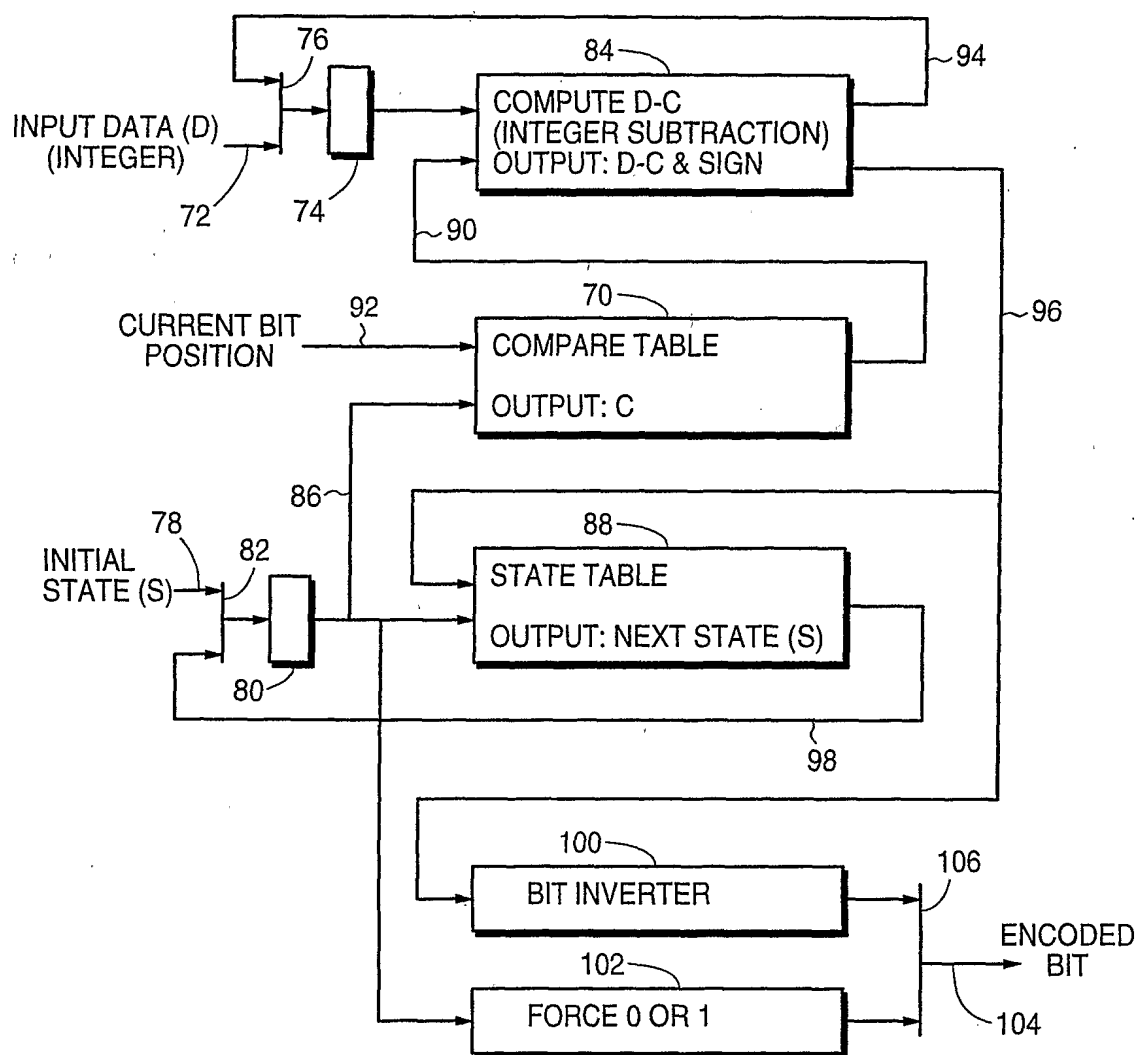


FIG. 7

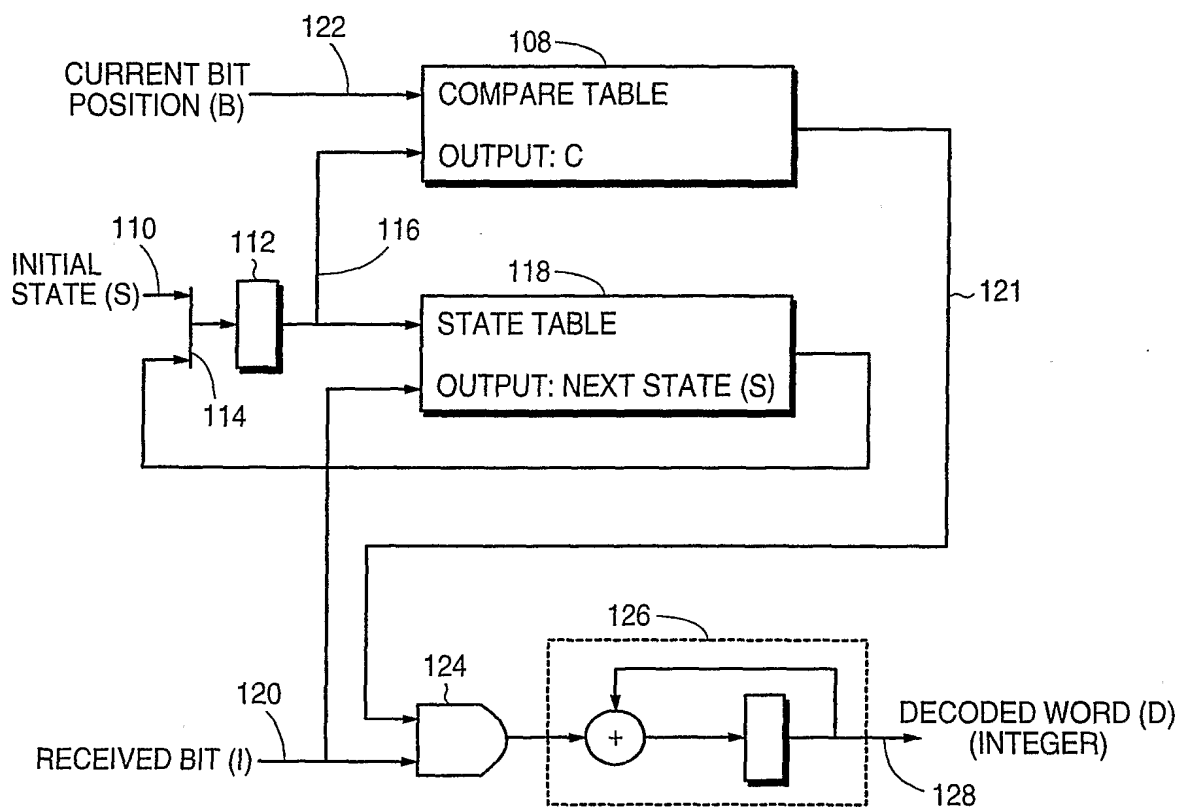
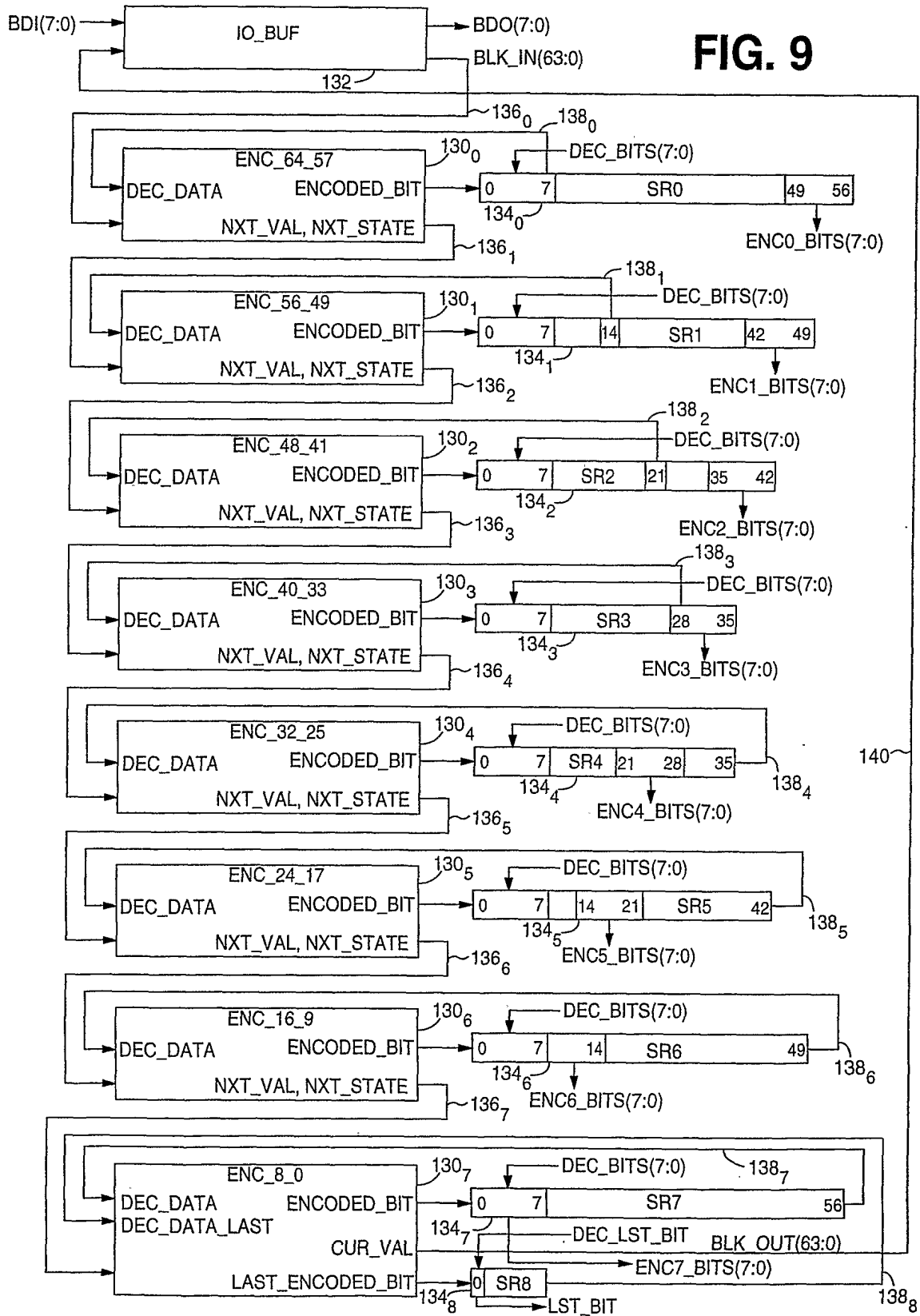
**FIG. 8**

FIG. 9



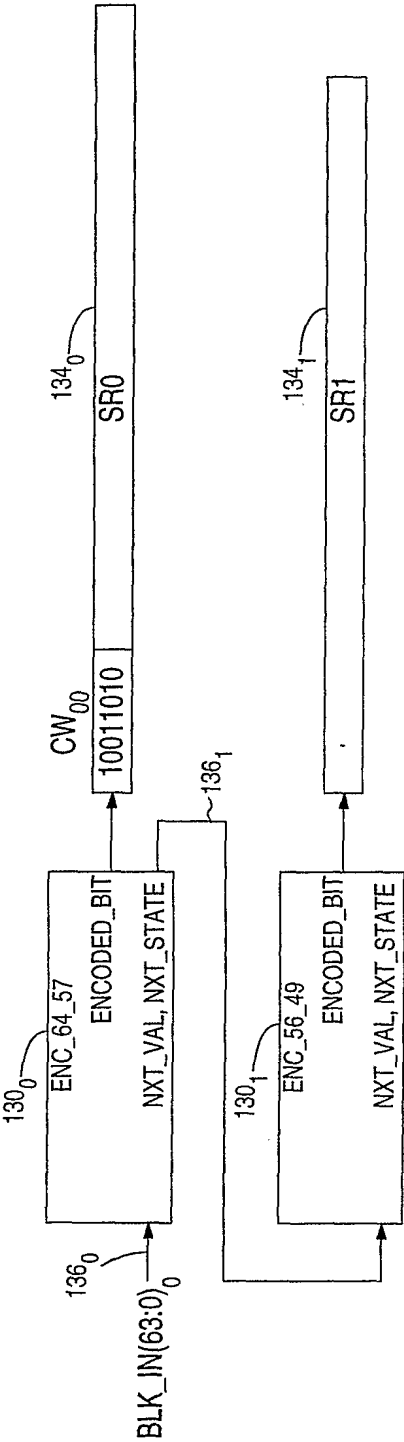


FIG. 10A

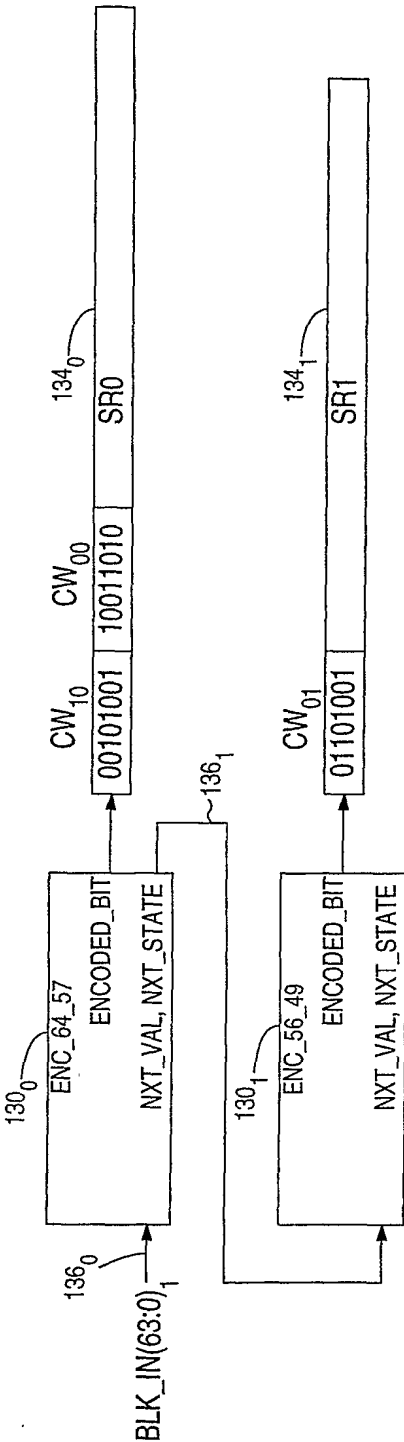


FIG. 10B

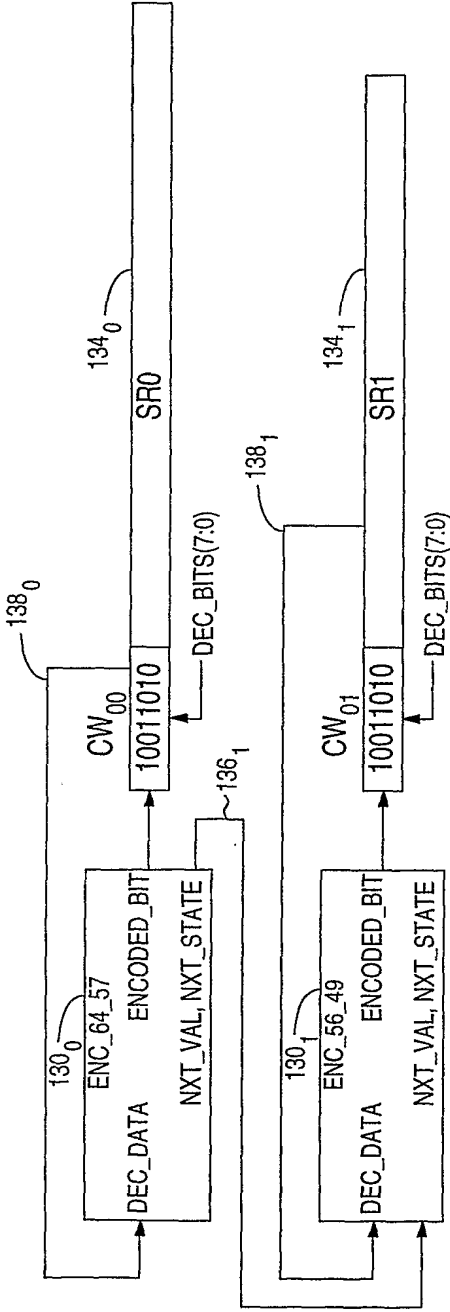


FIG. 11A

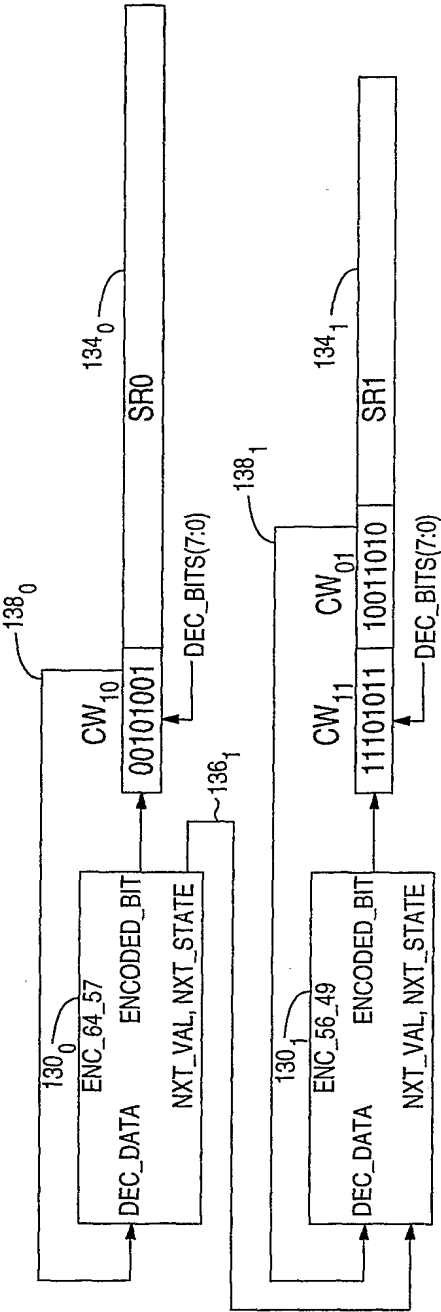


FIG. 11B

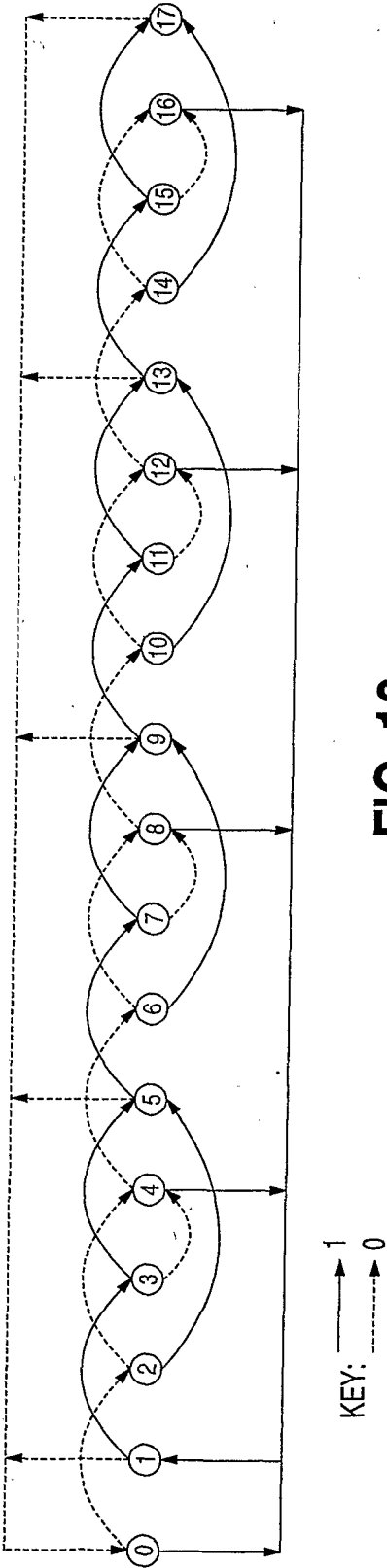


FIG. 12